Abstract. The final objective of an access control model is to provide a framework to decide if an action performed by subjects on objects is permitted or not. It is not convenient to directly specify an access control policy using concepts of subjects, objects and actions. In OrBAC (Organization Based Access Control), we can not only express static authorizations but also dynamic authorizations, depending on context. Formally, OrBAC is described in first order logic, where the context is one of the argument of predicate. We propose a new formalism based on description logic with defaults and exceptions \cite{1} to describe and reason on OrBAC model. This paper is an enrichment of a previous work \cite{10} with the introducing of an exception operator ($\epsilon$). This formalism covers not only concepts of information systems like users, objects, subjects and roles but also the context by the add of two operators of default ($\delta$) and exception ($\epsilon$). Notice that time complexity is still polynomial \cite{2}.

1 Introduction

Security policy models as DAC \cite{3}, MAC \cite{4,5} and RBAC \cite{6} (Role Based Access Cotrol) provide concepts that are useful in current IS. However, such security policies must also be adapted to deal with new requirements; rule are depending on context.

OrBAC (Organization Based Access Control) is useful to deal with some of these new requirements \cite{7}. OrBAC is a model that allows to express a security policy at organizational level, i.e., indepent from the implementation made for this policy. The access control policy does not directly apply to subject, object and action. It defines static authorizations that apply within organization to control the activities performed by roles on views. However, the OrBAC model also allows specification of more complex dynamic authorizations applying in a given context. The formalism used is the first order logic where the context is an additive argument.

The purpose of this paper is to present DL$_{\delta\epsilon}$-OrBAC which is a new formalization of OrBAC model based on description logic with defaults and exceptions ($\mathcal{ALC}_\delta\epsilon$) \cite{1}. Description Logic (DLs) \cite{8} are a family of knowledge used to describe and classify concepts and their instances. ($\mathcal{ALC}_\delta\epsilon$) extends the well known description logic ($\mathcal{ALC}$) \cite{8} with the two unary connectives ($\delta, \epsilon$) to express respectively default and exception.

The idea is to consider that when we are in a usual context, we obtain a default authorization and we express this fact with the connective ($\delta$), but if the context change, the authorization is excepted, and we express this exception with the connective ($\epsilon$). Notice, that even if we have more than exception, we can represent this fact with the
composition of operators of exception (ǫ), and in spite of this, time complexity still polynomial. This paper is an enrichment of a previous work [10] with the introducing of an exception operator (ǫ).

The rest of the paper is structured as follow. Section 2 presents OrBAC model, section 3 introduces description logic with defaults and exceptions. Section 4 defines $\text{DL}_{\delta\epsilon}$-OrBAC, shows how we express security and how we can infer access control rules in different contexts. We conclude in section 5 by the prospects of evolution of $\text{DL}_{\delta\epsilon}$-OrBAC.

## 2 OrBAC Model

The central entity in OrBAC model is Organization. An Organization can be seen as an organized group of subjects, each playing a specific role. In the medical domain, “Pierre and Marie Curie Center”, “Service of Pediatrics”, etc. are organizations. Subject, Action and Object are respectively abstracted into Role, Activity and View [7].

A Role is a set of Subjects to which the same security rule apply, for example, the subject “John” plays the role of “Doctor” in the organization “Service of Pediatrics”. A View corresponds to a set of Objects that satisfy a common property, for example, in the medical domain, the view “Medical record” corresponds to the object “Medical record of patient”. An Activity regroups Actions that partake of the same principle. In OrBAC model, Actions will mainly contain computer actions such as “read”, “write”, etc., while Activities contain “consulting”, “writing”, etc. Privileges only apply in specific contexts. Contexts can be used to specify the concrete circumstances where organizations grant roles permission to perform activities on views.

It considers that all actions which are not permitted are prohibited, so it suffice to defines only permission relation.

OrBAC is defined using eight basic sets of entities: OR (set of organizations), S (set of subjects), AC (set of actions), O (set of objects), R (set of roles), AV (set of activities), V (set of views) and C (set of contexts).

In the next section, we will introduce our formalism used to describe $\text{DL}_{\delta\epsilon}$-OrBAC which is based on description logic with defaults and exceptions.

## 3 Description Logic with Defaults and Exceptions

Description logic is actually largely used to represent concept hierarchies, it employs two kinds of formalisms for the knowledge representation: the terminological formalism (TBox) used to describe conceptual knowledge, the assertional formalism (ABox) used to allow facts to be stated [8].

In what follows, we present $(\mathcal{AL}_{\delta\epsilon})$, an extension of $\mathcal{AL}$ language with the two operators of defaults (δ) and exceptions (ε).

### 3.1 $\mathcal{AL}_{\delta\epsilon}$ Language

The description language with defaults and exception $\mathcal{AL}_{\delta\epsilon}$ is inductively defined from a set $\mathcal{R}$ of primitive roles and a set $\mathcal{P}$ of primitive concepts [9], augmented by the constant $\top$ (Top), with the abstract syntax rule:
the most general concept
| P             | primitive concept |
| C ∩ D        | concept conjunction |
| ¬P           | negation of primitive concept |
| ∀∃r : C      | C is a value restriction on all roles R(> 0) |
| δC           | default concept |
| Cε           | exception to the concept |

δ and ε are two unary connectives, ∩ is a binary conjunction connective and ∀∃ enables universal quantification on role values.

In the next section, we will give the formalization of OrBAC model based on description logic with defaults and exceptions. We will show how these two connectives (δ, ε) can be efficient to formalize access control.

4 DLδε-OrBAC

We now conceptualize the OrBAC model and construct a DL knowledge base capturing the characteristics of OrBAC, including the context with the use of (δ) and (ε).

4.1 The TBox

Given an OrBAC model, we define a DL knowledge base K, the alphabets of K includes the following atomic concepts: Organization, Subject, Object, Role, View, Action, Context and Activity. The TBox includes the following axioms, each axiom is illustrated with examples.

– Role Attribution Axiom: defines the relationship between subject and role.

| Subject ⊑ ⊤; Role ⊑ ⊤; Organization ⊑ ⊤ |
| Employ ⊑ EmployS.Subject ∩ EmployR.Role ∩ EmployOr.Organization |

Suppose that in a given hospital X, Jean is assigned in the role of Doctor, and Tom is assigned in the role of Surgeon. We express all these facts by the following rules.

Employ(E1) ⊑ EmployS.Subject(Jean) ∩ EmployR.Role(Doctor) ∩ EmployOr.Organization(X)
Employ(E2) ⊑ EmployS.Subject(Tom) ∩ EmployR.Role(Surgeon) ∩ EmployOr.Organization(X)

Where E1, and E2 are instances of Employ.

– View Definition Axiom: defines relationship between object and view.

| Object ⊑ ⊤; View ⊑ ⊤ |
| Use ⊑ UseO.Object ∩ UseV.View ∩ UseOr.Organization |

Suppose that in a given hospital X, Med-rec1 and ordinance1 are instances of concept Object, and Med-rec and ordinance are instances of concept View. We express all these facts by the following rules.

Use(U1) ⊑ UseO.Object(Med-rec1) ∩ UseV.view(Med-rec) ∩ UseOr.Organization(X)
Use(U2) ⊑ UseO.Object(Ordinance1) ∩ UseV.view(Ordinance) ∩ UseOr.Organization(X)
Where U1 and U2 are instances of Use.

– **Activity Definition Axiom**: defines relation between action and activity.

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action ⊑ T</td>
<td></td>
</tr>
<tr>
<td>Activity ⊑ T</td>
<td></td>
</tr>
<tr>
<td>Consider ⊑ ConsiderAc.Action ∩ ConsiderAv.Activity ∩ ConsiderOr.Organization</td>
<td></td>
</tr>
</tbody>
</table>

Suppose that in a given hospital X, action `write` is considered as a modification activity and action `read` as a consultation activity. We express all these facts by the following rules.

Consider(C1) ⊑ ConsiderAv.Activity(Modify) ∩ ConsiderAc.Action(write) ∩ ConsiderOr.Organization(X)
Consider(C2) ⊑ ConsiderAv.Activity(Consult) ∩ ConsiderAc.Action(read) ∩ ConsiderOr.Organization(X)
Where C1 and C2 are instances of Consider.

– **Context Definition Axiom**:

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context ⊑ T</td>
<td></td>
</tr>
<tr>
<td>Define ⊑ DefineAc.Action ∩ DefineS.Subject ∩ DefineO.Object ∩ DefineC.Context ∩ DefineOr.Organization</td>
<td></td>
</tr>
</tbody>
</table>

We need first to define the Normal context.

Define(D1) ⊑ DefineAc.Action(Write) ∩ DefineS.Subject(Jean) ∩ DefineO.Object(Ordinance) ∩ DefineC.Context(Normal) ∩ DefineOr.Organization(X)
Where D1 is an instance of Define.

– **Permission Attribution Axiom**: defines the relation between role, activity, view and context in an organization.

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permission ⊑ PermissionAv.Activity ∩ PermissionR.Role ∩ PermissionV.View ∩ PermissionC.Context ∩ PermissionOr.Organization</td>
<td></td>
</tr>
</tbody>
</table>

Every user who play the role of Doctor is permitted to modify an ordinace when the context normal is true.

Permission(P1) ⊑ PermissionAv.Activity(Modify) ∩ PermissionR.Role(Doctor) ∩ PermissionV.View(Ordinance) ∩ PermissionC.Context(Normal) ∩ PermissionOr.Organization(X)
Where P1 is an instance of Permission.

– **Hierarchy Definition Axiom**: defines the hierarchy between roles.

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub−role ⊑ Sub−role1.Role ∩ Sub−role2.Role ∩ Sub−roleOr.Organization</td>
<td></td>
</tr>
</tbody>
</table>

Role1 is a sub role of Role2 in organization Or
Suppose that in a given hospital X, a Surgeon play also the role of Doctor, then a surgeon inherits the set of authorizations of the role doctor. The next rule express this fact.

Permission(X, Doctor, Av, V, C) ⊑ Permission(X, Surgeon, Av, V, C)
And because we defined the concept of Sub-role, so the inheritance is expressed as follow:

\[
\text{Permission}(X, \text{Surgeon}, \text{Av}, V, C) \subseteq \text{Permission}(X, \text{Doctor}, \text{Av}, V, C) \cap \text{Sub} \cap \text{role}(X, \text{Surgeon}, \text{Doctor})
\]

- **Concrete Permission Axiom:**

  \[
  \text{Is} \cap \text{permitted} \subseteq \text{Is} \cap \text{permittedAc.Action} \cap \text{Is} \cap \text{permittedS.Subject} \\
  \cap \text{Is} \cap \text{permittedO.Object}
  \]

  Jean is permitted to write Diagnosis

  \[
  \text{Is} \cap \text{permitted}((I)) \subseteq \text{Is} \cap \text{permittedAc.Action(Write)} \\
  \cap \text{Is} \cap \text{permittedS.Subject(Jean)} \cap \text{Is} \cap \text{permittedO.Object(Diagnosis)}
  \]

**Definition of Rules of Security:**

Employ \cap Use \cap Consider \cap \delta \text{Permission} \cap \delta \text{Define} \subseteq \delta \text{Is} \cap \text{permitted} \\
Employ \cap Use \cap Consider \cap \text{Permission} \cap \text{Define}^c \subseteq \text{Is} \cap \text{permitted}^c

### 4.2 The ABox

The ABox of K includes eight catalogs of axioms: Organization assertions axiom, Subject assertions axiom, Object assertions axiom, View assertions axiom, Role assertions axiom, Action assertions axiom, Activity assertions axiom and Context assertions axiom.

In the next section, we show how a security policy can be modelized and how we can infer authorizations.

### 4.3 Inference and Subsumtion

- **Permission Hierarchy**

  We know that a Surgeon is a sub-role of Doctor, so we can write:

  \[
  \text{Sub} \cap \text{role}(S1) \subseteq \text{Sub} \cap \text{role1.ROLE(Surgeon)} \cap \text{Sub} \cap \text{role2.ROLE(Doctor)} \\
  \cap \text{Sub} \cap \text{roleOr.Organization(X)}
  \]

  When we want to know if a Surgeon is permitted to write an ordinance, we use the following rule:

  \[
  \text{Permission}(X, \text{Surgeon, Modify, Ordinance, Normal}) \subseteq \text{Permission}(X, \text{Doctor, Modify, Ordinance, Normal}) \cap \text{Sub} \cap \text{role}(S1)
  \]

  and then the answer in this case is **Yes** because we use the inheritance of properties in the normal case.

- **Access Control if Normal Context is True**

  Within organization X, Normal context holds between subject Jean, action Write and object Diagnosis1, we obtain D2, an instance of concept Define.

  \[
  \text{Define}(D2) \subseteq \text{DefineAc.Action(Write)} \cap \text{DefineS.Subject(Jean)} \\
  \cap \text{DefineO.Object(Diagnosis1)} \cap \text{DefineC.Context(Normal)} \\
  \cap \text{DefineOr.Organization(X)}
  \]

  We also know that Diagnosis1 is an object used in a view Diagnosis, an instance U3 of Use is write as:
And finally, we know that in organization X, each person who play the role of Doctor is permitted to modify Diagnosis, when Normal context is true, we write an instance P2 as:

- Permission(P2) ⊑ PermissionAv.Activity(Modify)
- PermissionR.Role(Doctor) ⊑ PermissionV.View(Diagnosis)
- PermissionC.Context(Normal) ⊑ PermissionOr.Organization(X)

Now, the question is: Is Jean permitted to write diagnosis in a normal context? We have:

- Jean play role of doctor in organization X: Employ(E1);
- and, Diagnosis1 is an object used in the view Diagnosis: Use(U3);
- and, Write is considered as a modification activity: Consider(C1);
- and, by default, within organization X, context Normal holds between subject Jean, action Write and object Diagnosis1: δDefine(D2);
- and finally, by default, in organization X, each person who plays the role of Doctor is permitted to modify Diagnosis, when Normal context is true: δPermission(P2).

Using security rules, we can deduce that the precedent proposition subsume δIs permitted(I1).

And because Is – permitted(I1) ⊑ δIs – permitted(I1), we can deduce that Jean is permitted to write diagnosis.

---

**Access Control if Context “Contamination-risk” is True.** We have within organization X, context Contamination-risk holds between subject Jean, action Write and object Diagnosis1, we obtain D3, an instance of concept Define.

- Define(D3) ⊑ DefineAc.Action(Write) ⊑ DefineS.Subject(Jean)
- DefineO.Object(Diagnosis1) ⊑ DefineC.Context(Contamination-risk) ⊑ DefineOr.Organization(X)

We know that context Contamination-risk is an exception of context normal, DefineC.Context(Contamination-risk) ≡ (DefineC.Context(Normal))

If we substitute DefineC.Context(Contamination-risk) by its value, we obtain:

- δDefine(D3) ⊑ δDefineAc.Action(Write) ⊑ δDefineS.Subject(Jean)
- δDefineO.Object(Diagnosis1) ⊑ δDefineC.Context(Normal)
- δ(DefineC.Context(Normal)) ⊑ δDefineOr.Organization(X)

Using the rule A' ≡ δA ⊑ A', we obtain:

- (DefineC.Context(Normal)) ⊑ δDefineC.Context(Contamination-risk)

after replacement, we get:

- δDefine(D3) ⊑ δDefineAc.Action(Write) ⊑ δDefineS.Subject(Jean)
- δDefineO.Object(Diagnosis1) ⊑ (DefineC.Context(Normal))
- δDefineOr.Organization(X)

which means that: Define(D2) ⊑ δDefine(D3)

In the context Contamination-risk, we create a new instance P3 which is defined as follow:
Permission(P3) ⊑ PermissionAv.Activity(Modify)
¬PermissionR.Role(Doctor) ⊓ PermissionV.View(Diagnosis)
¬PermissionC.Context(Normal) ⊓ PermissionC.Context(Contamination-risk) ⊓ PermissionOr.Organization(X)

Context Contamination-risk is an exception of context normal

PermissionC.Context(Contamination-risk) ≡ (PermissionC.Context(Normal))^¢

After substitution, we get:
¬Permission(P3) ⊑ PermissionAv.Activity(Modify)
¬PermissionR.Role(Doctor) ⊓ PermissionV.View(Diagnosis)
¬PermissionC.Context(Normal) ⊓ (PermissionC.Context(Normal))^¢
¬PermissionOr.Organization(X)

We know that, A^¢ ≡ δA △ A', we obtain:
Permission(P3) ⊑ PermissionAv.Activity(Modify)
¬PermissionR.Role(Doctor) ⊓ PermissionV.View(Diagnosis)
¬(PermissionC.Context(Normal))^¢ ⊓ PermissionOr.Organization(X)

So, we can deduce: Permission(P2)^¢ ⊑ δPermission(P3)

The question we can ask is: Is Jean permitted to write Diagnosis when context Contamination-risk is true?

We have:
- Jean play role of doctor in organization X: Employ(E1);
- and, Diagnosis1 is an object used in the view Diagnosis: Use(U3);
- and, Write is considered as a modification activity: Consider(C1);
- and, by default, within organization X, context Contamination-risk holds between subject Jean, action Write and object Diagnosis1: δDefine(D3);
- and finally, by default, in organization X, each person who plays the role of Doctor is permitted to modify Diagnosis, when context Contamination-risk is true: δPermission(P3).

We obtain: Employ(E1) △ Use(U3) △ Consider(C1) △ δPermission(P3) △ δDefine(D3)
≡ Employ(E1) △ Use(U3) △ Consider(C1) △ δDefine(P3)^¢
¬δDefine(D2)^¢
≡ Employ(E1) △ Use(U3) △ Consider(C1) △ Permission(P2)^¢ △ δDefine(D2)^¢

Using security rules, we can deduce that the precedent proposition subsume Is − permitted(I1)^¢. And because Is − permitted(I1) △ Is − permitted(I1)^¢, then we can’t deduce Is-permitted(I1), and Jean is not permitted to write diagnosis when there is a contamination risk.

5 Conclusions

The aim of this paper is to give a logical formalisation to DLm−OrBAC model using the expressive Alδ language. We showed how default and exceptional knowledge are well suited to represent and reason about access control, we illustrate this fact with a small example of medical information system.

To implement and reason about this model, we need to choose a reasoner which take account default and exceptional knowledge. There are many tools for the classical description logic [11], we mention Classic, Fact++, RacerPro, but in our knowledge, there is no tool for description logic with default and exception reasoner.
The next step of our work is to develop such a reasoner. \textit{Neo–Classic\textsubscript{hi}} is actually our preoccupation.

References